Bruno Canale
Bruno Giordano
Fábio T. Sancinetti
Wanderson Ferreira

Atividade Final

PSI5886 Princípios de Neurocomputação
Professor: Emílio Del Moral Hernandez

Atividade Final: Redes Neurais Convolucionais

Universidade de São Paulo
Escola Politécnica
Programa de Pós-Graduação em Engenharia Elétrica
PPGEE

São Paulo - Brasil
07 de dezembro de 2016

Lista de abreviaturas e siglas

CNN	*Convolutional Neural Networks* - Redes Neurais Convolucionais
GPU	*Graphic Processor Unit* - Unidade de Processamento Gráfico
MLP	*Multilayer Perceptron* - Perceptron em muti-camadas
RNA	Redes Neurais Artificiais
RNN	*Recurrent Neural Networks* - Redes Neurais Recorrentes

Sumário

1 Introdução

Quisque facilisis auctor sapien. Pellentesque gravida hendrerit lectus. Mauris rutrum sodales sapien. Fusce hendrerit sem vel lorem. Integer pellentesque massa vel augue. Integer elit tortor, feugiat quis, sagittis et, ornare non, lacus. Vestibulum posuere pellentesque eros. Quisque venenatis ipsum dictum nulla. Aliquam quis quam non metus eleifend interdum. Nam eget sapien ac mauris malesuada adipiscing. Etiam eleifend neque sed quam. Nulla facilisi. Proin a ligula. Sed id dui eu nibh egestas tincidunt. Suspendisse arcu.

2 LIPSUM

2.1 Wolverinis Adamantium

Sed mattis, erat sit amet gravida malesuada, elit augue egestas diam, tempus scelerisque nunc nisl vitae libero. Sed consequat feugiat massa. Nunc porta, eros in eleifend varius, erat leo rutrum dui, non convallis lectus orci ut nibh. Sed lorem massa, nonummy quis, egestas id, condimentum at, nisl. Maecenas at nibh. Aliquam et augue at nunc pellentesque ullamcorper. Duis nisl nibh, laoreet suscipit, convallis ut, rutrum id, enim. Phasellus odio. Nulla nulla elit, molestie non, scelerisque at, vestibulum eu, nulla. Ut odio nisl, facilisis id, mollis et, scelerisque nec, enim. Aenean sem leo, pellentesque sit amet, scelerisque sit amet, vehicula pellentesque, sapien.

(LISA LAB, 2016)

Exemplo de citação em linha segundo Rojas (1996) aqui.

3 Visualização do reconhecimento em uma Rede Neural

3.1 Base de dados: MNIST

A base de dados MNIST (*Mixed National Institute of Standards and Technology*) por **??**), é uma base de dados que contem 60000 exemplos de digitos manuscritos para treino e outros 10000 exemplos para treino disponível em <http://yann.lecun.com/exdb/mnist/>.

As imagens dos dígitos possuem 28x28 pixels, entretanto os dígitos em si possuem 20x20, são normalizados e centralizados.

Segundo **??**), algumas técnicas de reconhecimento de dígitos foram aplicadas a esses bancos como classificadores lineares, KNN (K-nearest neighbors), entre outras, obtendo taxa de erro entre 12% e 0.54% respectivamente. Entretanto, como o objetivo deste trabalho é auxiliar o entendimento dos funcionamento das redes convolucionais através da visualização dos resultados, os valores de taxa de erro mais interessantes são os relacionados a redes convolucionais similares as apresentadas nesse trabalho. **??**) apresenta redes convolucionais de LeNet-1, LeNet-4, LeNet-5 com taxas de erro de 1.7%, 1.1% e 0.7% respectivamente. **??**) ainda apresenta os resultados para outras redes convolucionais com medida de erros como *cross-entropy* com taxas de erro de 0.6%,

O resultado obtido pela rede explorada neste capítulo deste trabalho obteve taxa de erro de 0.94% na base de teste do MNIST e sua estrutura será detalhada nas próximas sessões.

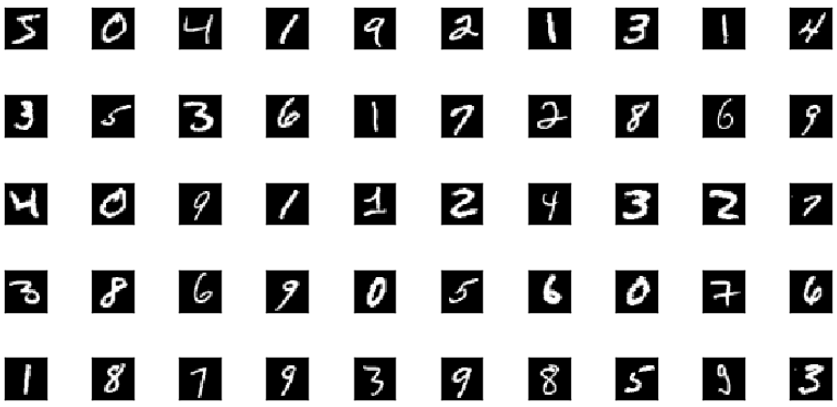


Figura 1

Figura 2

3.2 Rede Convolucional para reconhecimento de dígitos manuscritos do MNIST

A rede a ser apresentada neste trabalho é composta da camada de entrada e 16 camadas para o processamento. Com o objetivo de simplificar a visualização, as camadas de convolução e ativação que são consecutivas foram agrupadas em um único bloco.

A identificação mostrada a seguir é feita após o treinamento da rede com a arquitetura exibida na imagem **??**.

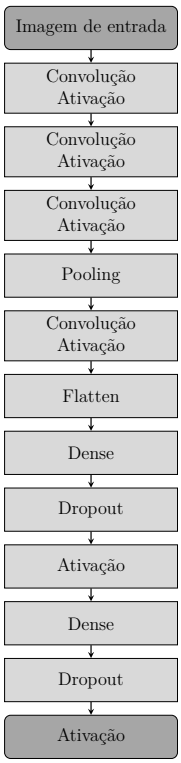


Figura 3 – Arquitetura da rede CNN utilizada neste trabalho.

A implementação da rede descrita pela imagem **??** é exibida no código abaixo:

```
from keras.layers import Activation
from keras.layers.convolutional import Convolution2D, MaxPooling2D
from keras.layers import Flatten, Dense
from keras.layers.core import Dropout

dropout_rate=0.3

modelo = Sequential()

def keras_add(m,op):
        m.add(op)
```

```
        l = m.layers[-1]
        print l.name, l.get_output_shape_at(-1)

keras_add(modelo,Convolution2D(32, 2, 2,
        border_mode='valid', input_shape=(28, 28, 1)))
keras_add(modelo,Activation('relu'))

keras_add(modelo,Convolution2D(8, 12, 12))
keras_add(modelo,Activation('relu'))

keras_add(modelo,Convolution2D(16, 8, 8))
keras_add(modelo,Activation('relu'))

keras_add(modelo,MaxPooling2D(pool_size=(2, 2)))

keras_add(modelo,Convolution2D(32, 2, 2))
keras_add(modelo,Activation('relu'))

keras_add(modelo,Flatten())
keras_add(modelo,Dense(100))
keras_add(modelo,Dropout(dropout_rate))
keras_add(modelo,Activation('relu'))

keras_add(modelo,Dense(numero_classes))
keras_add(modelo,Dropout(dropout_rate))
keras_add(modelo,Activation('softmax'))
```

O objetivo das camadas convolucionais com ativação no início dessa rede é de encontrar características dos números e reduzir parcialmente o tamanho da imagem inicial antes do primeiro *pooling*. Para melhor compreensão dos efeitos da convolução nas imagens foram utilizados em cada camada convolucional filtros dos tamanhos: 32 filtros de 2x2, 8 filtros de 12x12, 16 filtros de 8x8 e 32 filtros de 2x2.

A visualização dos filtros obtidos após o treinamento foi obtida da seguinte forma:

```
img = np.zeros((28,28))
for a in xrange(28):
  for b in xrange(28):
    img[a][b] = X_treino[0][a,b]
    for i in xrange(len(modelo.layers)):
      m0 = modelo.layers[i]
```

```
name = m0.name
if "convolution" not in name:
  continue
ws = m0.get_weights()

flt = np.zeros((np.shape(ws[0])[0],np.shape(ws[0])[1]))
fig=plt.figure(figsize=(8,8))
for oz in xrange(np.shape(ws[0])[2]):
  for z in xrange(np.shape(ws[0])[3]):
    ax = fig.add_subplot(6, 6, z+1)
    sa = np.shape(ws[0])[0]
    sb = np.shape(ws[0])[1]
    for a in xrange(sa):
      for b in xrange(sb):
        flt[a][b] = ws[0][a,b,oz,z]
        plt.xticks(np.array([]))
        plt.yticks(np.array([]))
        plt.tight_layout()
        ax.imshow(flt,interpolation='nearest', cmap=plt.cm.gray)
```

Os filtros descritos acima são visualizados na imagem **??**.

Após as primeiras camadas de convolução foi adicionada uma camada de *pooling* para auxiliar na eliminação de ruídos e reduzir o número de elementos a serem explorados pela rede.

A seguir, uma nova camada de convolução foi adicionada para detectar novas características novamente. Por conta da estrutura do *framework* do Keras, utilizado neste trabalho é necessário antes da rede *fully-connected* ou (*dense*) transformar a entrada em um vetor unidimensional, representada mais a frente por uma fita.

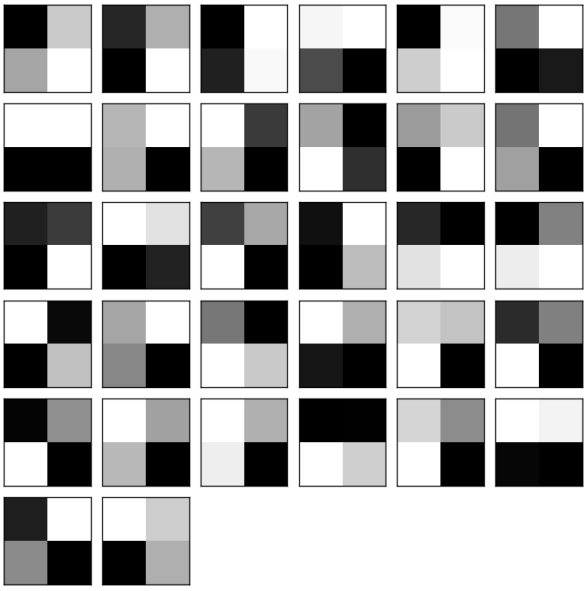
A camada *dense* trabalha como uma camada de redes neurais conectando todos os elementos da entrada com todos os elementos da saída, visando estabelecer a relação da importância entre os elementos ativados e os resultados obtidos.

O próximo elemento é uma camada de *Dropout* com objetivo de minimizar a dependência de algumas características específicas dos dígitos. Em seguida é utilizada uma camada de ativação para evidenciar os elementos mais importantes encontrados após a camada *dense+droupout*.

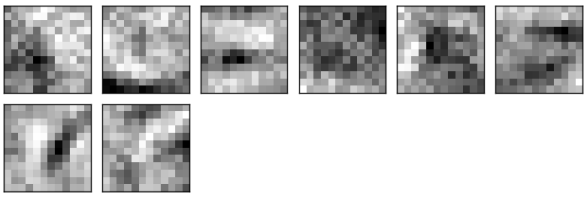
Os últimos dois passos são novamente uma camada *dense*, agora reduzindo as entradas de 100 para 10 elementos (correspondentes aos dígitos de 0 a 9) e uma nova camada de ativação agora com função *softmax* para ativação de um único elemento

(representando a escolha da rede do elemento mais provável identificado pela rede).

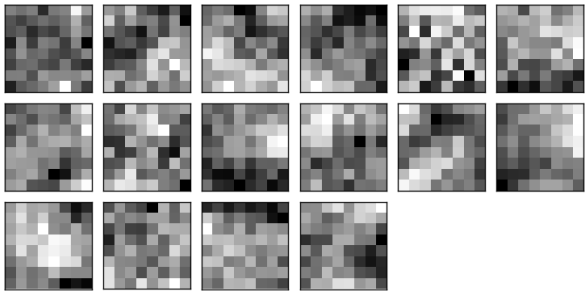
A identificação do dígito é iniciada com a introdução da imagem contendo o item a ser identificado. Neste exemplo será utilizado o número 0 apresentado na imagem 5.



(a) Filtros 2x2 - primeira convolução

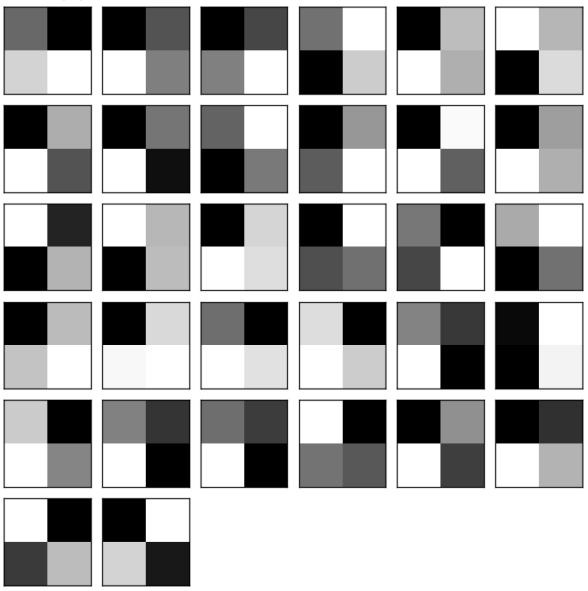


(b) Filtros 12x12 - segunda convolução



d

(c) Filtros 8x8 - terceira convolução



(d) Filtros 2x2 - terceira convolução

Figura 4 – Filtros convolucionais obtidos do treinamento.

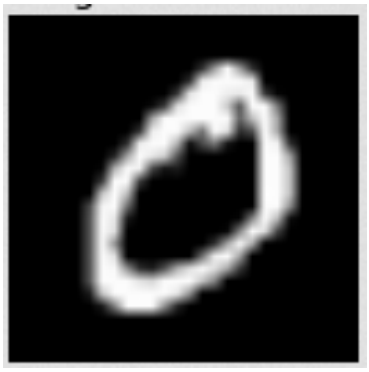
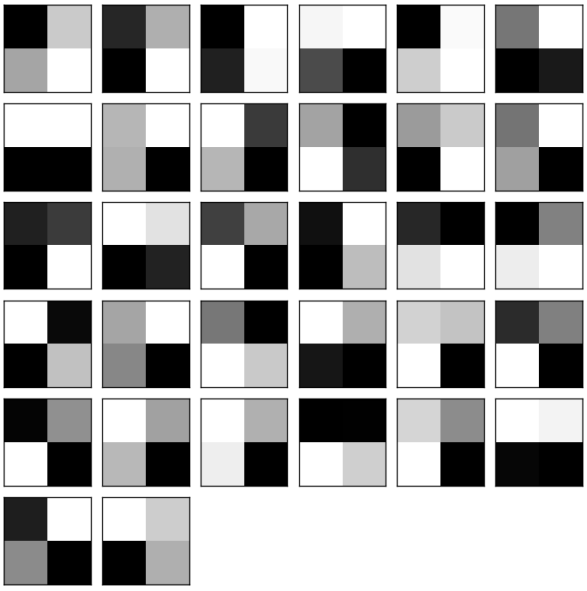
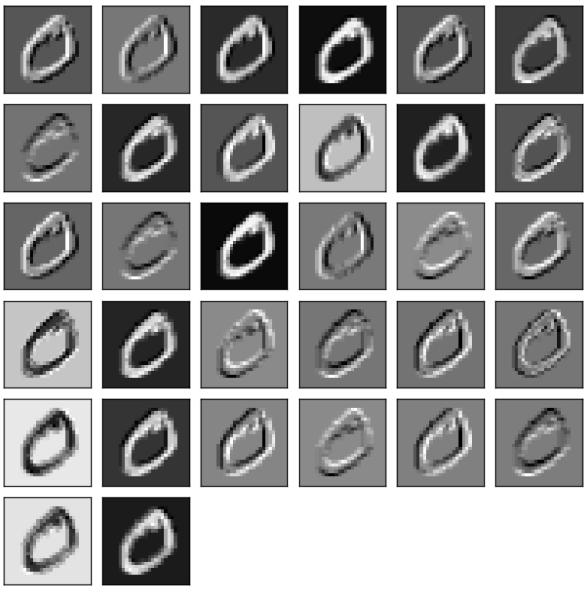


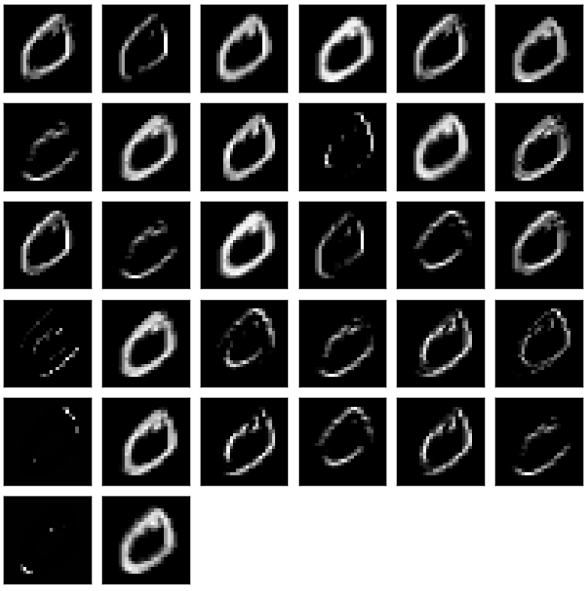
Figura 5 – Número 0 escrito a mão disponível na base de dados do MNIST.



(a) Filtros 2x2 aplicados a primeira camada de convolução

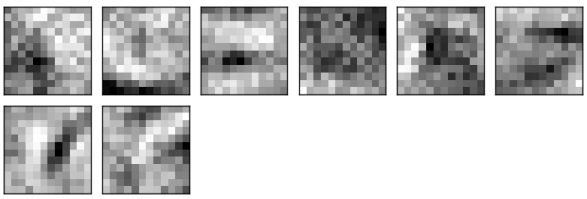


(b) Convolução 1

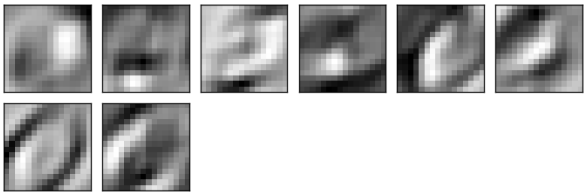


(c) Ativação

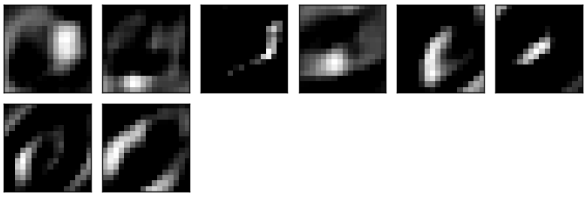
Figura 6 – Primeiro resultado da identificação no primeiro par de camadas de convolução e ativação.



(a) Filtros 12x12 aplicados a segunda camada de convolução

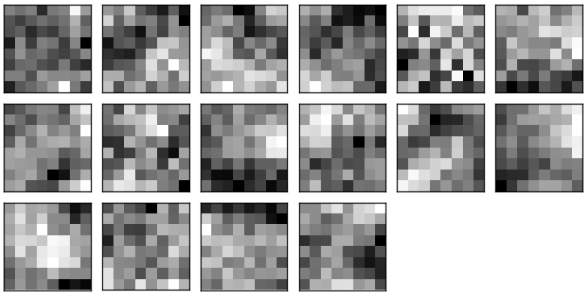


(b) Convolução 2

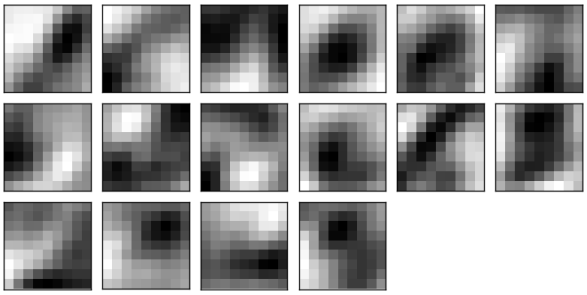


(c) Ativação

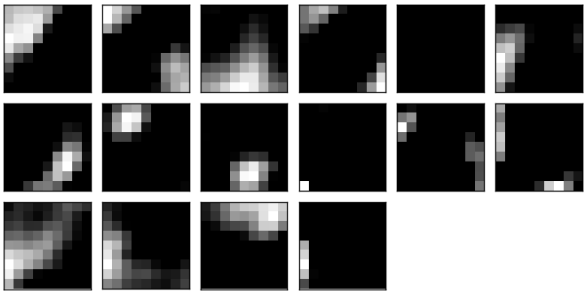
Figura 7 – Primeiro resultado da identificação no segundo par de camadas de convolução e ativação.



(a) Filtros 8x8 aplicados a terceira camada de convolução



(b) Convolução 3



(c) Ativação

Figura 8 – Primeiro resultado da identificação no terceiro par de camadas de convolução e ativação.

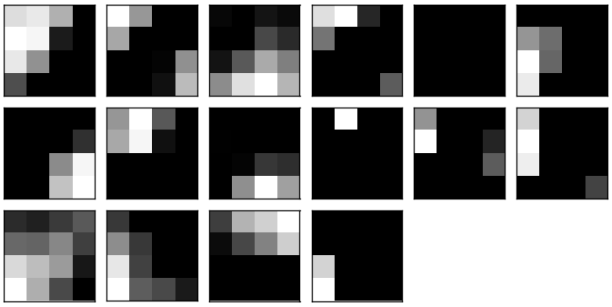
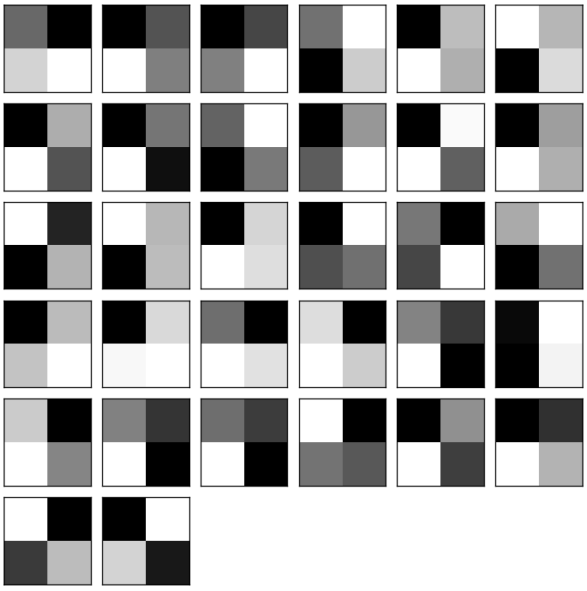
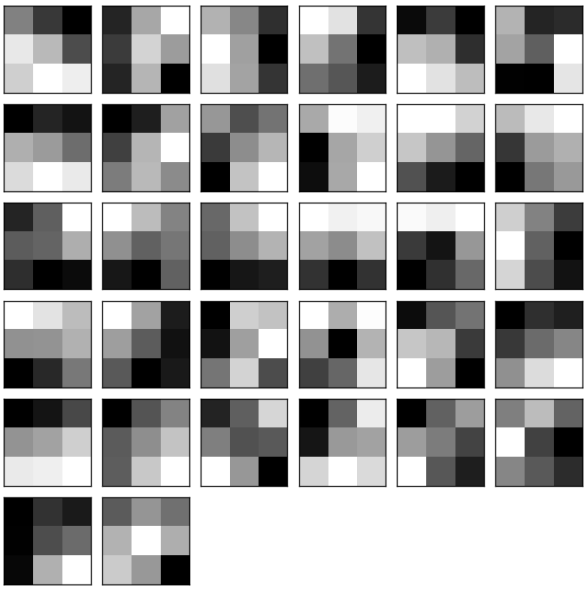


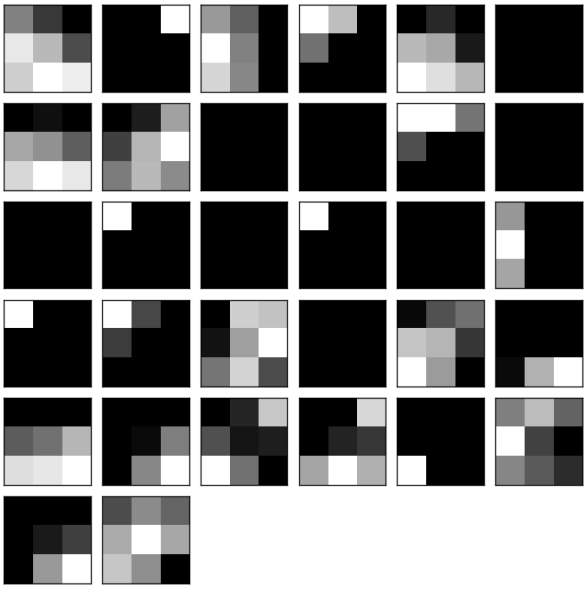
Figura 9 – Resultado da camada de Pooling



(a) Filtros 2x2 aplicados a quarta camada de convolução



(b) Convolução 4



(c) Ativação

Figura 10 – Primeiro resultado da identificação no quarto par de camadas de convolução e ativação.

Figura 11 – Resultado da camada de Flatten

Figura 12 – Resultado da camada de Flatten

Figura 13 – Resultado da camada de ativação antes da saída

Figura 14 – Resultado da camada de ativação antes da saída

Figura 15 – Resultado da camada de ativação para definição da saída

4 Conclusão

Sed mattis, erat sit amet gravida malesuada, elit augue egestas diam, tempus scelerisque nunc nisl vitae libero. Sed consequat feugiat massa. Nunc porta, eros in eleifend varius, erat leo rutrum dui, non convallis lectus orci ut nibh. Sed lorem massa, nonummy quis, egestas id, condimentum at, nisl. Maecenas at nibh. Aliquam et augue at nunc pellentesque ullamcorper. Duis nisl nibh, laoreet suscipit, convallis ut, rutrum id, enim. Phasellus odio. Nulla nulla elit, molestie non, scelerisque at, vestibulum eu, nulla. Ut odio nisl, facilisis id, mollis et, scelerisque nec, enim. Aenean sem leo, pellentesque sit amet, scelerisque sit amet, vehicula pellentesque, sapien.

Referências

LISA LAB. *DeepLearning 0.1 documentation*. 2016. Disponível em: <http://deeplearning.net/tutorial/mlp.html>. Citado na página 5.

ROJAS, R. *Neural Networks - A Systematic Introduction*. 1996. Disponível em: <http://page.mi.fu-berlin.de/rojas/neural/index.html.html>. Citado na página 5.